Global Robot Localization using Iterative Scale Invariant Feature Transform

Hashem Tamimi and Andreas Zell
Computer Science Dept., University of Tübingen,
Sand 1, 72076 Tübingen, Germany,
{tamimi, zell}@informatik.uni-tuebingen.de

ABSTRACT

We propose an approach that can reduce the feature extraction time of the Scale Invariant Feature Transform (SIFT). The main idea is to search for the keypoints around a set of randomly generated particles rather than to perform exhaustive search in the whole difference of Gaussian pyramid. The proposed approach makes it possible to define the required number of keypoints in advance. A relation between the number of required features and the feature extraction time can be drawn. The matching time of two sets of keypoints is consequently minimized. To demonstrate its performance, the approach is applied to the robot global localization problem. The results are successful, with much less keypoints and less computation time than the original SIFT.

1 INTRODUCTION

The problem of robot localization can be classified as either global or local localization [10]. In global localization, the robot tries to discover its position without previous knowledge about its location. In local localization, the robot must update its position using its current data from its sensors as well as the previous information that it has already accumulated. The lack of any historical information about its surroundings makes global localization more challenging [10].

Vision based robot localization demands image features with many properties. On the one hand the features should exhibit invariance to scale and rotation as well as robustness against noise and illumination. On the other hand they should be extracted very quickly so as not to hinder the other tasks that the robot plans to perform. Local descriptors are commonly employed in robot localization because they can be computed efficiently, are resistant to partial occlusion, and are relatively insensitive to changes in viewpoint.

There are two considerations to using local descriptors [4]: First, the interest points should be localized in position and scale. Typically, interest points are placed at local peaks in a scale-space search, and filtered to preserve only those that are likely to remain stable over transformations. Second, a description of the interest point is built; ideally, this description should be distinctive (reliably differentiating one interest point from others), and invariant over transformations caused by changes in camera pose and lighting. While the localization and description aspects of interest point algorithms are often designed together, the solutions to these two problems are independent [8].

SIFT features, explained in details in section 2, have been widely used in the robot localization field. In [11] the SIFT scale and orientation constraints are employed for matching stereo images; after matching the features they used a least-squares procedure to reach better localization problem. In [1] a modified version of the SIFT approach is proposed and used to solve the global robot localization; their approach takes the properties of panoramic images into consideration. The work in [12] proposes an approach to modeling the poses-dependent characteristics of the SIFT, their model is a learning based one and is successfully applied to the robot localization. In order to further minimize the classification errors during localization the work in [5] has proposed extracting SIFT features from each image and then using spatial relationships among the locations by means of a hidden Markov model. In [3] an image map based on SIFT and Harris corners is built and used later for localization.

The remaining sections of this paper are organized as follows: Section (2) is a review of the SIFT approach. Section (3) introduces the idea behind the iterative SIFT and presents its algorithm. Section (4) deals with the application of our approach to the problem of global robot localization and presents the similarity measure needed for successful localization. In section (5) the computational time of SIFT is compared with that of iterative SIFT and the advantages and disadvantages of both approaches are discussed. Section (6) gives the experimental results of applying the iterative SIFT to the robot localization. Finally we conclude this paper in section (7).

2 SCALE INVARIANT FEATURE TRANSFORM

The Scale Invariant Feature Transform (SIFT) developed by Lowe [7] is invariant to image translation, scaling, ro-

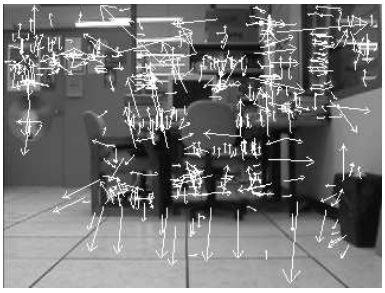

Fig. 1. A sample image which has 320×240 pixels and 181 keypoints.

Fig. 2. A sample image which has 320×240 pixels and 254 keypoints.

tation, and partially invariant to illumination changes and affine for 3D projection. The SIFT algorithm has 4 major stages:

1. Scale-space extrema detection: The first stage searches over scale space using a Difference of Gaussian function to identify potential interest points.
2. Keypoint localization: The location and scale of each candidate point is determined and keypoints are selected based on measures of stability.
3. Orientation assignment: One or more orientations are assigned to each keypoint based on local image gradients.
4. Keypoint descriptor: A descriptor is generated for each keypoint from local image gradients information at the scale found in stage 2.

An important aspect of the algorithm is that it generates a large number of features over a broad range of scales and locations. The number of features generated is dependent on image size and content, as well as algorithm parameters. Figures 1 and 2 show two images of the same size but the number of features, illustrated using arrows, is nearly doubled.

The SIFT feature algorithm is based upon finding locations within the scale space of an image which can be reliably extracted. The first stage finds scale-space extrema located in $D(x, y, \sigma)$, the Difference of Gaussians (DoG)

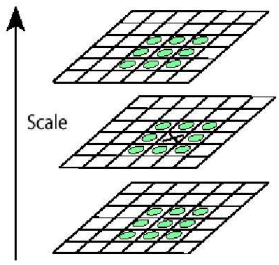

Fig. 3. Local extrema detection, the pixel marked $\times$ is compared against its 26 neighbors in a $3 \times 3 \times 3$ neighborhood that spans adjacent DoG images (from [7]).

function, which can be computed from the difference of two nearby scaled images separated by a multiplicative factor k:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \quad (1)$$

where $L(x, y, \sigma)$ is the scale space of an image, built by convolving the image $I(x, y)$ with the Gaussian kernel $G(x, y, \sigma)$. Points in the DoG function which are local extrema in their own scale and one scale above and below are extracted as keypoints. Generation of extrema in this stage is dependent on the frequency of sampling in the scale space k and the initial smoothing σ_0. The keypoints are then filtered for more stable matches, and more accurately localized to scale and subpixel image location using methods described in [9].

Before a descriptor for the keypoint is constructed, the keypoint is assigned an orientation to make the descriptor invariant to rotation. This keypoint orientation is calculated from an orientation histogram of local gradients from the closest smoothed image $L(x, y, \sigma)$. For each image sample $L(x, y)$ at this scale, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ is computed using pixel differences:

$$\begin{aligned} m(x, y)^2 &= (L(x+1, y) - L(x-1, y))^2 \\ &+ (L(x, y+1) - L(x, y-1))^2 \end{aligned} \quad (2)$$

$$\theta(x, y) = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \quad (3)$$

The orientation histogram has 36 bins covering the 360 degree range of orientations. Each point is added to the histogram weighted by the gradient magnitude, $m(x, y)$, and by a circular Gaussian with variance σ that is 1.5 times the scale of the keypoint. Typical keypoint descriptors use 16 orientation histograms aligned in a 4×4 grid. Each histogram has 8 orientation bins each created over a support window of 4×4 pixels. The resulting feature vectors are

128 elements with a total support window of 16×16 scaled pixels. For a more detailed discussion of the keypoint generation and factors involved see [9].

3 ITERATIVE SIFT

SIFT features are distinctive invariant features used to robustly describe and match digital image content between different views of a scene. While invariant to scale and rotation, and robust to other image transforms, the SIFT feature description of an image is typically large and slow to compute.

For example, the work in [2] is a study of SIFT features for outdoor robot localization. Although his approach was able to pick up features which stay in place despite the varying illumination, the authors reported some disadvantages of using SIFT, specifically the time it takes to extract the features from an image is long. Further, the number of features is immense, which poses problems when searching for the patching pairs, along with having to store a large amount of data.

The main objective of the iterative SIFT approach is to reduce the number of keypoints and their corresponding extraction and matching time, while maintaining the same descriptor for each keypoint. In the classical SIFT approach, keypoints are detected by testing each value in the DoG at each scale with the 8 surrounding values of the same scale as well as with 9 neighbouring values in the scale above and 9 neighbouring values in the scale below. The first and last DoG scales are not examined. This means $26 \times m \times n$ comparisons for a DoG of size $m \times n$, taking into consideration that points around a given border of each DoG are not included in the keypoint detection, as seen in figure 3 . Since SIFT establishes multiple scales in each octave, the above analysis is applied several times to each scale in each octave. Each octave has one quarter of the pixels of the previous one, so that keypoint detection in lower octaves requires more time than in higher ones. We aim to modify this exhaustive search into a sample based one.

In the proposed approach, the number of keypoints can be defined in advance. The process of finding the keypoints continues iteratively without the need for sequentially going through the whole scale space. This involves two phases. The first phase is randomly searching the scale space for local extrema. The random search is followed by an update phase only when the local extremum is more likely to be found. The theory behind the iterative SIFT approach is mainly based on the assumption that local extrema points are located in a blob region [6], i.e. smooth wide two dimensional hills or valleys. In other words, blobs are regions in the image that are either significantly brighter or significantly darker than their surroundings. A local extremum cannot be located on a flat region and can hardly be found near it. Another possible location of local extrema are spikes, i.e. rapidly changing narrow regions. But since the scale space structure involves multiple smoothing operations on the image, only information on the coarse scale remains and the spikes are filtered out.

Algorithm 1 The iterative SIFT algorithm.

Definitions:
$NSamples$: The number of samples.
$NKeys$: The number of requested keypoints.
$NTrials$: The number of trials.
$NScales$: The number of Scale images.
initialization: $Keypoints \leftarrow \{\}$;
for $Scale \leftarrow 1$ to $NScales$ **do**
  $\{(x_i, y_i)\} \leftarrow$ RandomCoord(), given that $isBlob\,(p\,(x_i, y_i))$ is True, $\forall i = 1, 2, ..., Nsamples$;
  $i \leftarrow 0$;
  while $i < NSamples$ And $|Keypoints| < NKeys$ **do**
    $i \leftarrow i + 1$;
    $trial \leftarrow 0$;
    $ExtremaFound \leftarrow False$;
    while $trial < NTrials$ And $\neg ExtremaFound$ **do**
      if $isExtremum\,(x_i, y_i)$ **then**
        $Key_i \leftarrow KeypointDescriptor(x_i, y_i)$;
        $Keypoints \leftarrow Keypoints \cup \{Key_i\}$;
        $ExtremaFound \leftarrow True$;
      else
        $(x_i, y_i) \leftarrow findNewCandidate\,(x_i, y_i)$;
      end if
      $trial \leftarrow trial + 1$;
    end while
  end while
end for

With the above assumption we can say that our search mechanism involves dealing only with two cases when searching for a local extremum: 1) In the case where we detect a blob region, an update phase handles the search for the position of the local extremum in that region. The search ends either when the local extremum is found or when a given number of trials elapses. 2) In the case where we detect a non-blob region, the result of the search in this area is ignored and the search is started somewhere else.

The test whether a point p lies in a blob region or not is shown as a function $isBlob()$ in equation (4), where T_{Scale} is a threshold depending on the scale of the point p.

$$isBlob = \begin{cases} True & (Abs(p) > T_{Scale}) \\ False & otherwise \end{cases} \quad (4)$$

In the iterative SIFT algorithm, see algorithm (1), the following functions are used: The function *KeypointDescriptor()* builds the keypoint descriptor for the point located in

(x_i, y_i). The function *RandomCoord()* returns a set of randomly chosen (x,y) coordinates bounded by the size of the current scale. The function *isExtremum()* tests whether the point located in (x_i, y_i) is a local maximum or local minimum. This involves comparing the value of a point with the 26 neighbours as explained above. The function *findNewCandidate()* searches the 8 neighbours of the point (x_i, y_i) in the same scale and returns the coordinate of the maximum point if $p(x_i, y_i)$ is positive and the coordinate of the minimum point if $p(x_i, y_i)$ is negative. Note that the DoG image contains both positive and negative points with the median 0.

The algorithm is clarified as follows: We first initialize a set of samples with random numbers, each of which holds a value that represents the coordinate of one of the points in the current scale. The samples are then verified so that only those that have a value above the given threshold T_{Scale} remain. This reflects our assumptions that a value above this threshold is most probably a point that lies in a blob. The total number of samples in the algorithm after the verification should be $NSamples$ and depends on the size of the scale image. After some initial experiments, we found that setting $NSamples = r \times M \times N$ with $r = 3$ leads to best results. Here (M, N) are the size of the current scale image. The search for the keypoints in the algorithm is applied to one scale after another. The search stops when the required number of keypoints $NKeys$ is reached. The search within a blob involves testing whether the current point is a local extremum using the function *isExtremum()*. If the point is an extremum then the keypoint descriptor is generated for that point. If the current keypoint is not a local extremum, the function *findNewCandidate()* returns a new coordinate for a candidate local extremum. With this update, the sample is assumed to move in the direction of the local extrema. The update goes on until the local extremum is found or until the maximum number of trials, $NTrials$, is reached. In the later case the point is neglected and the search is performed by another sample in another new location.

4 GLOBAL ROBOT LOCALIZATION

Now we apply the iterative SIFT approach to the problem of global localization and compare the performance of it with the SIFT approach in terms of accuracy and computation time.

4.1 Problem Definition

Vision based global localization can be generally considered as an image retrieval problem [13], where features of the images that the robot encounters during its navigation are compared with those stored in the database of the robot. Unlike the local localization problem, here the robot is assumed to retrieve its location without considering any historical information about the position of the robot. For this reason the problem of global localization is also referred to as the kidnapped robot problem [10].

The localization in mainly performed in two phases: First, the robot performs an exploration phase during, which it discovers the environment for its first time, collects images from different positions and extracts features from these images. The features are stored in the robot memory along with the corresponding robot positions, usually in (x, y, θ) terms. The computation power that the robot requires during the exploration phase is not critical.

When the robot performs the localization phase, the robot should retrieve its position by comparing the features from its current image with the features in the database. The robot should perform this as fast as possible so as not to delay the other jobs which the robot intends to perform.

When applying iterative SIFT to the robot localization, we try to reduce the computational effort of the feature extraction as much as possible while maintaining high localization accuracy. This means that the robot will try to localize itself using less keypoints than the classical SIFT. The keypoints in iterative SIFT are found through a random process. This makes it dangerous that those keypoints found in the exploration phase are different from those found in the localization phase or that the common keypoints between the two are not sufficient for localization. Since the computation time of the exploration phase is not critical, we overcome this problem by applying classical SIFT in the exploration phase and iterative SIFT in the localization phase.

4.2 Similarity Measure

When comparing images through their corresponding features we apply the following similarity measure between each two images: For each keypoint in a given image k_a (with 128 elements) we find the two closest matched keypoints k_b and k_c from the other image. The matches are calculated through the squared distance measure in equation (5):

$$d\left(k_x, k_y\right) = \sum_{i=1}^{128} \left(k_{x_i} - k_{y_i}\right)^2 \tag{5}$$

A positive match of the keypoint k_a with k_b is recognized if $4 * d\left(k_a, k_b\right) < d\left(k_a, k_c\right)$. This leads to robust matching. The final decision which image is similar to which is then given by the one with the maximum number of positive matches.

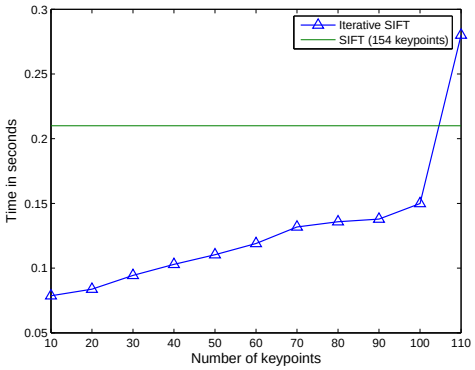

Fig. 4. The time required to find a given number of keypoints using iterative SIFT versus the time of SIFT.

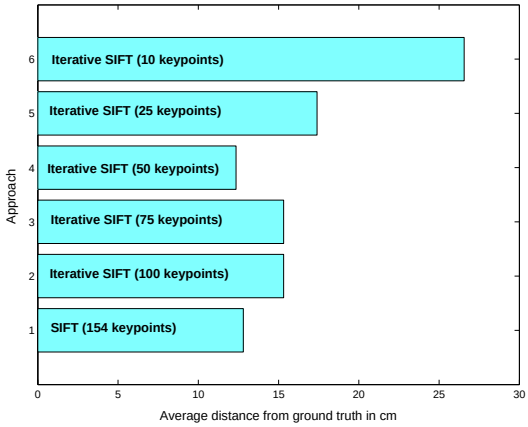

Fig. 5. Average distance from the ground truth.

5 TIME CONSIDERATIONS OF ITERATIVE SIFT

Before introducing the experimental results, we first demonstrate the ability of iterative SIFT to extract a given number of keypoints and how this varies with the required time. Figure 4 consists of repeated experiments with the given keypoints and the corresponding time. The plot also shows that the SIFT approach finds a constant number of keypoints and requires a relative high and constant time.

It is worth noting that even though iterative SIFT finds a good amount of features in less time than it would take SIFT, our approach fails to keep doing so as the number of given keypoints gets higher and is not able find the number of keypoints that the classical SIFT finds within less or equal time. This is because our approach is based on particles which are randomly distributed in the search space. These particles need much more time than the linear approach of SIFT in order to cover the whole search space.

Still, the application of robot localization, as well as other similar applications does not need such a large number of keypoints as SIFT produces, as seen in section (6).

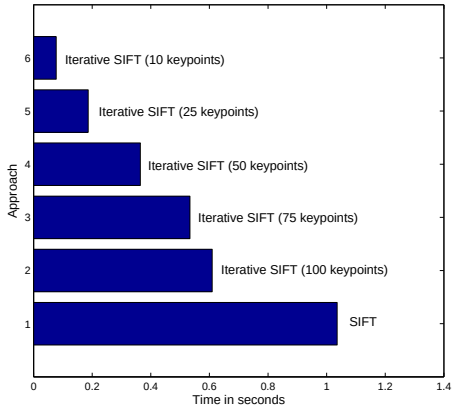

Fig. 6. Average time to match the keypoints during localization.

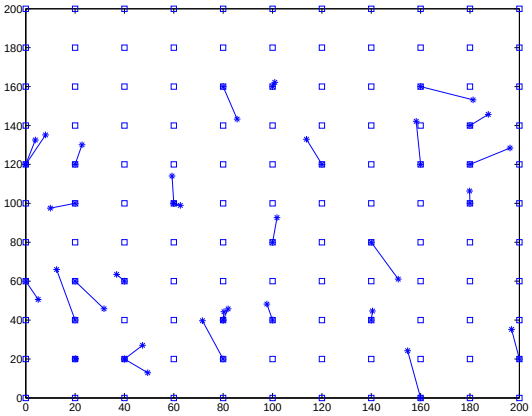

Fig. 7. Robot pose estimates and corresponding ground truth using SIFT. Average distance =12.81 cm.

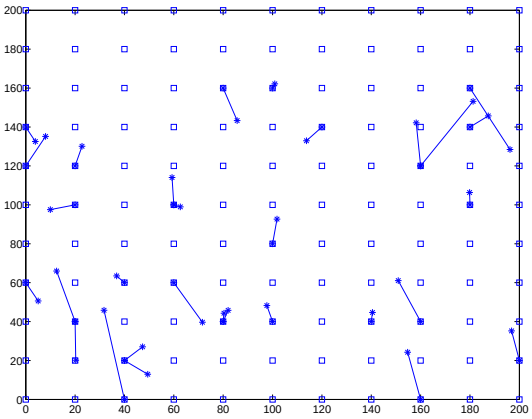

Fig. 8. Robot pose estimates and corresponding ground truth using iterative SIFT with 50 keypoints. Average distance = 12.35 cm.

6 EXPERIMENTAL RESULTS

To simulate the global localization of a mobile robot we made the following experiments, which consist of trying to

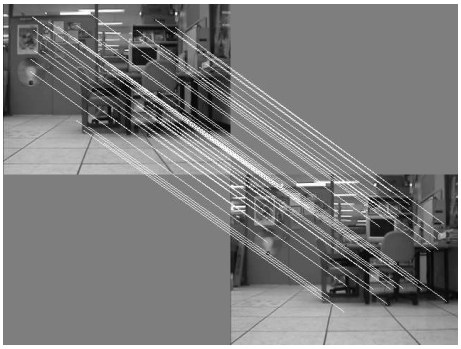

Fig. 9. Matching two images using the SIFT approach. There are 57 common keypoints. The distance between the positions of the two images is 10.17 cm.

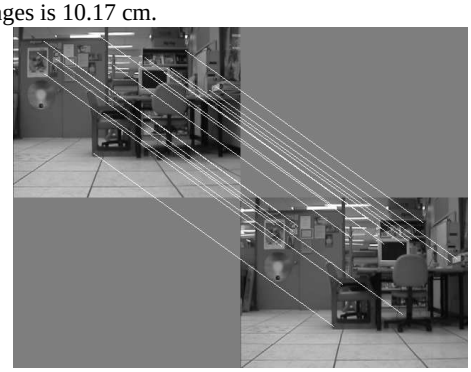

Fig. 10. Matching two images using the iterative SIFT approach with 50 keypoints. There are 17 common keypoints. The distance between the positions of the two images is 10.17 cm.

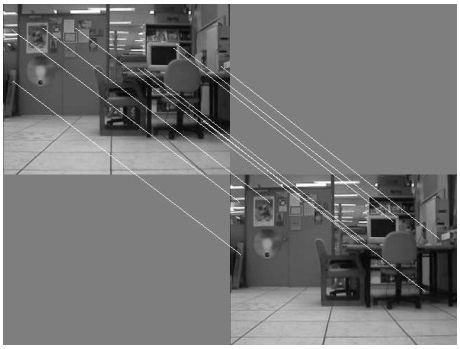

Fig. 11. Matching two images using the iterative SIFT approach with 25 keypoints. There are 11 common keypoints. The distance between the positions of the two images is 18.1 cm.

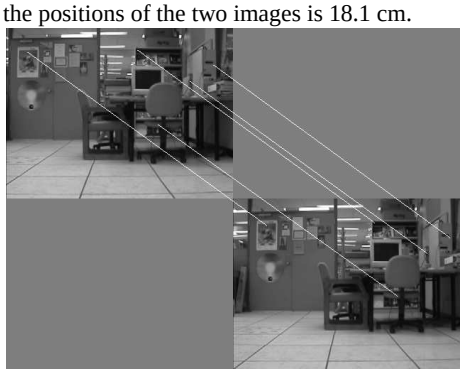

Fig. 12. Matching two images using the iterative SIFT approach with 10 keypoints. There are 5 common keypoints. The distance between the positions of the two images is 14.9 cm.

match a set of indoor images taken by a mobile robot during its localization phase to another set of images which represents the exploration phase. Each of the images is associated with ground truth information, namely the robot position in (x, y, θ) terms. An accurate localization is then judged to be the one which minimize the distance between the positions of the images being compared from the two sets. We use the same image set as in [12], which consists of 121 indoor images. The images are taken in a 11×11 grid in a robot lab, 20 cm apart from each other. For testing, another 30 images distributed in the robot lab are used. The camera in all the experiments is always headed towards the same angle, which means that we deal only with (x, y) coordinates and can neglect the orientation θ.

During the exploration phase we apply the classical SIFT approach to extract all the keypoints from the 121 images. Then we use the iterative SIFT approach to extract the keypoints from the 30 test images and match them with the keypoints of the exploration phase.

We have seen that the iterative SIFT can be applied with a small amount of time, but this should be compromised with the localization accuracy. Figure 5 illustrates the localization accuracy in terms of the average distance between the robot positions. This distance is to be minimized for better localization. The figure shows the results of both SIFT and iterative SIFT using different numbers of keypoints. It is worth mentioning that this experiment was repeated many times to test the different random distributions of the particles and consequently to possibly find different keypoints. We have noticed that the final matching results did not change as we repeated the experiments. This is because of the robustness in the keypoints and the similarity measure.

Figure 5 also shows that using an iterative SIFT with 50 keypoints worked even better than SIFT itself and requires less time. Although this result can not be generalized it shows that the many features that the SIFT produces do not always lead to more accurate localisation.

Since the number of keypoints in the iterative SIFT is clearly reduced, the time for matching two sets of keypoints during the robot localization, as explained in subsection (4.2), can be minimized to a great extent. Figure 6 shows the time required by each approach.

In figure 7 a map is illustrated, the squares are the image positions during the exploration phase, and the stars are the image positions in the localization phase. The lines between them are the matching results when applying SIFT features. Matching is accomplished as discussed in section

(4.2). When repeating the same experiment using iterative SIFT with 50 keypoints as seen in figure 8 we can see that the localization results are better than with SIFT.

Figures 9, 10, 11 and 12 demonstrate the localization of an image using SIFT and iterative SIFT of different number of keypoints. The common keypoints between the images are also illustrated.

7 CONCLUSION

In this paper we have introduced a practical idea to speed up the SIFT approach. The number of keypoints can be defined in advance and the computation time is proportional to that given number of keypoints. When applying the approach to the global robot localization problem, we demonstrated that this approach is suitable since not many keypoints are needed. The approach can be generally applied to any similar problem. It should be obvious that any modification to the original SIFT approach, such as in the keypoint descriptor or orientation assignment, may also be applied to this approach.

ACKNOWLEDGMENT

The authors would like to thank Prof. David Lowe for providing them with the source code of SIFT features. The first author would also like to acknowledge the financial support by the German Academic Exchange Service (DAAD) of his Ph.D scholarship at the University of Tübingen.

REFERENCES

[1] H. Andreasson and T. Duckett. Topological localization for mobile robots using omni-directional vision and local features. In *Proceedings IAV 2004, the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, 2004.

[2] M. Artač and A. Leonardis. Outdoor mobile robot localisation using global and local features. In Danijel Skočaj, editor, *Computer vision - CVWW '04 : proceedings of the 9 th Computer Vision Winter Workshop*, pages 175–184, Piran, February 2004. Slovenian Pattern Recognition Society.

[3] C. Silpa-Anan R. Hartley. Localisation using an image-map. In *Proceedings of the 2004 Australasian Conference on Robotics and Automation*, Canberra, Australia, 2004.

[4] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *CVPR (2)*, pages 506–513, 2004.

[5] J. Kosecka and F. Li. Vision based topological Markov localization. In *IEEE International Conference on Robotics and Automation (ICRA 2004)*, pages 1481–1486, New Orleans, USA, 2004.

[6] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[8] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. Submitted to PAMI, 2004.

[9] P. Muir. Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 25–31. Springer-Verlag, New York, 1990.

[10] S. Se, D. Lowe, and J. Little. Local and global localization for mobile robots using visual landmarks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 414–420, Maui, Hawaii, October 2001.

[11] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2051–2058, Seoul, Korea, May 2001.

[12] R. Sim and G. Dudek. Learning generative models of invariant features. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3481–3488, Sendai, Japan, 2004.

[13] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *Proc. IEEE Int. Conf. Robotics & Automation (ICRA)*, 2002.